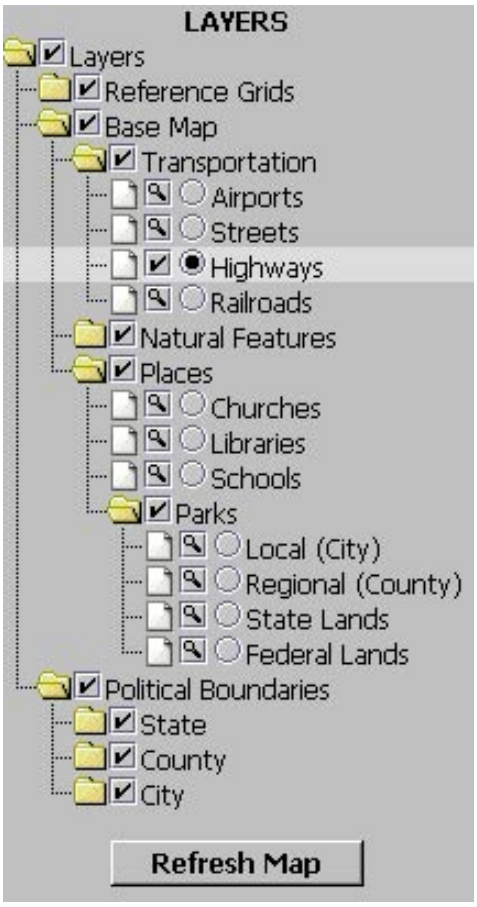
dbGroupToc Documentation

version 1.4 01/27/2003 David Bollinger (davebollinger@hotmail.com)

Introduction



dbGroupToc is a grouping Table Of Contents (or "Layer List") for the ArcIMS HTML Viewer. It is capable of grouping any number of layers into any number of groups which may be collapsed and expanded. It supports visibility toggling at both the layer and group levels. It supports a single "swatch" per layer to indicate legend rendering, as well as an optional "legend" per layer which can include detailed legend rendering if desired. It distinguishes between layers which are completely visible, and layers which have been flagged visible but are not visible at the current zoom level. It is fairly browser-independent, requires only minimal changes to existing code to implement, and is clearly broken into code and data modules.

Considerations

This code allows for considerable flexibility in the layout of the layout list: the number of groups, the number of layers per group, the combined legend functionality, presenting groups and layers in any desired order, etc. However, this flexibility comes at a price: the legend is essentially static. It is not intended for use with a site that employs a lot of dynamic data, especially dynamic renderers. Rather it is intended for sites with a known set of data which can be carefully laid out ahead of time. It might be possible, with considerable coding effort, to make this legend behave more dynamically, but that is not its intended purpose.

This code was initially designed for the ArcIMS 3.1 HTML Viewer code set in mind, but has also been tested with the 3.0 and 4.0 code sets. It has been tested with recent releases of MS and Netscape browsers on the Windows platform. It has not been thoroughly tested on older browsers, and is known to not work on NS 4.7x and earlier. It has not been tested on other platforms including Mac and Unix.

Installation

1. Copy all files into a subdirectory within your web site's directory. You may place this code in any directory you wish, but these instructions and the sample data assume you will use a subdirectory named "dbGroupToc". If you decide to use a different directory name you will need to amend these instructions for your particular installation.

2. Replace the default toc.htm file, located in your web site's directory, with the toc.htm included herein. Note that these instructions assume you have not otherwise modified the default toc.htm. If your toc.htm has already been customized then you will have to amend these instructions for your particular installation.

3. Include the dbGroupToc JavaScript in MapFrame.htm either before or after the other included JavaScript files:

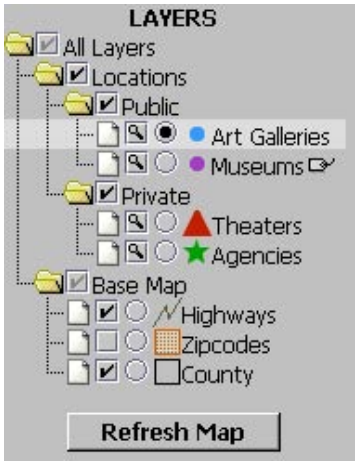
```
// at around line 30 of MapFrame.htm . . .
      </SCRIPT>
      <!-- dbGroupToc -->
      <SCRIPT TYPE="text/javascript" LANGUAGE="JavaScript" SRC="dbGroupToc/dbgtCode.js"></SCRIPT>
      <SCRIPT TYPE="text/javascript" LANGUAGE="JavaScript" SRC="dbGroupToc/dbgtData.js"></SCRIPT>
      <!-- Basic Map Display -->
      <SCRIPT TYPE="text/javascript" LANGUAGE="JavaScript" SRC="ArcIMSparam.js"></SCRIPT>
// etc . . .
```

4. Customize the dbgtData.js file for your particular site. Examine the sample definitions provided and see below for further detailed instructions.

5. That's it!

Samples

The included dbgtData.js contains a set of definitions designed to work with the "SanFrancisco" sample Map Service provided with ArcIMS. To use this sample, create the Map Service and a basic HTML Viewer site for the Map Service as described in the ArcIMS documentation. Then install the dbGroupToc modifications as described above. (*Note that this legend is for demonstration only, and does not necessarily reflect the actual contents or intent of that map service.*) Once properly set up you should see:



Usage Notes

- This code is intended for programmers. No apologies are given for anything that I've forgotten to document! You have the code, go figure it out! :-D *(It is assumed that almost every site will desire some degree of customization.)*

- The paths and image filenames specified in the resource object must combine to form a valid URL. You may alternately leave the path variables blank and fully specify the path and filename in the filename variable. This might be useful, for example, if your swatch images were located in several different directories.

- Items are added to the toc in a top-down order. The order in which the items are displayed will match the order in which they are added. Items are also added to individual groups in a top-down order. This was done so that the code for toc definition more closely resembles the actual toc display, to ease coding. Contrast this with the typical GIS software implementation of "adding a layer" which usually implies adding it to the top of the existing list. (for example, AXL files, which read from the bottom-up)

- Not all layers in the AXL have to be added to the legend. However, any layers not added to the legend will require some other mechanism to toggle their visibility or set them active, if so desired.

- Group visibility is implemented as a tri-state condition. If all contained layers are hidden, then the group is considered hidden. If all contained layers are visible, then the group is considered visible. If there is a combination of visible and hidden layers, then the group is considered to be in a mixed state. The default order of state change is that mixed becomes visible. This could be changed so that mixed becomes hidden by altering the value passed to setVisible() within group.writeHTML().

- Creating multiple toc's is entirely supported. Though, I admit, I haven't yet found a good use for it! But if you do have some need for switching toc's "on-the-fly" it would be possible to create them all in advance, then add code to toc.htm to switch between which one of them to write out.

- To create swatches or legends I suggest you simply do a screen grab of the toc from either Author or ArcExplorer and paste the result into your favorite image editing software for further tweeking.

- Note that changes to visibility settings immediately update the internal tracking variables. What this means is that if a user changes a layer's visibility setting, even if they neglect to press the "Refresh Map" button, those visibility settings will take effect upon the next map refresh however it may be initiated. Contrast this with the behaviour of the default layer list which requires pressing the "Refresh Map" button to update layer visibility status. This behaviour is intentional.

Customization Notes

How do I change the background color of the TOC?
Look in dbgtStyle.css for the BODY element and change the background-color value. Other HTML formatting can be controlled from this file as well, such as font family, font size, margins, etc.

How do I change the color of the active layer highlight or eliminate it?
Look in dbgtStyle.css for the TR.Highlight element and change the background-color value to another color, or set it to transparent.

How do I implement multiple TOC's?
In dbgtData.js define multiple instances of the TOC object…

```
var tocOne = new TOC("TOC ONE");
// add groups and layers to tocOne
var tocTwo = new TOC("TOC TWO");
// add groups and layers to tocTwo
var toc = tocOne;  // a reference to the currently active TOC object
```

Then, to change TOC's, simply change the reference somewhere within your code:

```
// the notation "t.toc" is appropriate if this code were located in toc.htm,
// otherwise you may need to adjust the parent reference as appropriate
t.toc = tocOne; // or…
t.toc = tocTwo; // then…
t.toc.refresh();
```

How do I do <whatever> when the layer's swatch is clicked?
See the comments in the function TOC_onSwatchClick in dbgtCode.js. By default, clicking the swatch performs no action. It is up to you to implement any specific desired action in response to this event.

How do I display layer info or metadata when the layer name is clicked?
See the comments in the function TOC_onNameClick in dbgtCode.js.

How do I toggle labelling for a layer?
See the comments in the function TOC_onLabelClick and ITEM_toggleLabel. The code for the toggle button itself is implemented, but it is up to you to implement the actual changes to the renderers necessary to enable/disable labels.

How do I create a group where only one layer can be visible at a time?
Add a boolean property to the GROUP object called isSingleVisible (for example) and set that value true for the desired group. Then within TOC_onVisibleClick change the existing code as such:

```
  else if (item instanceof LAYER) {
    if (item.parent.isSingleVisible)
      item.parent.setVisible(false);  // hide all layers within this group
    item.toggleVisible();  // show only this single layer
  }
```

How do I make this TOC work with Netscape 4.76 (for example, or earlier)?
Voodoo and/or witchcraft may be required.

Also, search the code for "customization notes:" to find other miscellaneous hints and tips.

Class Reference

RESOURCE

About: A "wrapper" for all shared settings used by the other objects. This was done to prevent littering the global namespace. The code expects a global singleton object of this class named "dbgtResource", as shown in the sample data. The name of this variable cannot be easily changed. Although presented in the data module, it is not expected that you will need to change this object much, except perhaps to alter the paths.

Properties:

Numerous. See sample data for a full listing and description.

Methods:

RESOURCE(name)

Constructor. Returns an object of class RESOURCE.
"name" is a string identifier. Not currently used for anything.

LAYER

About: A representation of a single layer (theme) within the legend. LAYER is descended from ITEM and contains all of ITEM's properties and methods.

Properties:

index – a integer containing the array index used by the HTML Viewer to identify this layer.
Internally maintained.
swatch – a string containing a URL-formatted filename of a swatch image.
If this value is the empty string ("") then no swatch image will be displayed.
legend – a string containing a URL-formatted filename of a legend image.
If this value is the empty string ("") then no legend image will be displayed.
labelField – a string containing the field name used to label this layer.
If this value is the empty string ("") then no label icon will be displayed.
labelled – a boolean indicating the current label state.

Methods:

LAYER(name,swatch,legend,labelField)

Constructor. Returns an object of class LAYER.
"name" is a string identifier. Used for labelling. **Must match layer name in AXL**.
"swatch" is a string containing the filename of a swatch image.
If swatch is null or the empty string ("") then no swatch is displayed.
"legend" is a string containing the filename of a legend image.
If legend is null or the empty string ("") then no legend is displayed.
"labelField" is a string containing the field name used to label this layer.
If labelField is null or the empty string ("") then no label icon is diplayed.

init()

Initializes internal values. Called internally.

writeHTML()

Returns a string containing the contents of the layer in HTML format.

GROUP

About: A representation of a single group within the legend. A group may contain any number of items. GROUP is descended from ITEM and contains all of ITEM's properties and methods.

Properties:

parent – a reference to the TOC object which contains this group.
name – a string identifier. Used for labelling.
opened – a boolean indicating the current opened/closed state.
items – an array of GROUP and/or LAYER objects.

Methods:

GROUP(name,opened)

Constructor. Returns an object of class GROUP.
"name" is a string identifier. Used for labelling. Must match layer name in AXL.
"opened" is a boolean value indicating the initial opened/closed state of the group. If true, group will initially be opened.

init()

Initializes internal values. Called internally.

addItem(item)

Adds an item to the items[] array. Returns the item.
"item" is a reference to a GROUP or LAYER object.

addGroup(group)
addLayer(layer)

Deprecated methods for backward compatibility.
Functionally equivalent to addItem(item)

writeHTML()

Returns a string containing the contents of the group in HTML format.
Will cascade and call writeHTML() for each LAYER object in the layers[] array.

ITEM

About: A "template" class of which both GROUP and LAYER are descendents. The ITEM class contains methods that apply to both GROUP's and LAYER's, thus most methods are recursive.

Properties:

parent – a reference to the object that contains this item.
tocid – an integer that uniquely identifies an item within the toc.
name – a string identifier. Used for labelling.

Methods:

ITEM(obj,name)

Pseudo-Constructor. Modifies an existing object to resemble an ITEM.
"obj" is a reference to an object of type GROUP or LAYER.
"name" is a string identifier. Used for labelling.

findItemByTocID(tocid)

Returns a reference to the item identified by tocid, or null if not found.
"tocid" is an integer that uniquely identifies an item within the toc.

setActive()

Sets the active state of an item to true.
Currently setActive is only supported for LAYER objects.

toggleLabel()
Toggles the labelled state of an item.

getVisible()
Returns the visible/hidden state of an item.
The GROUP tri-state condition returns the value 99999.

setVisible(value)
Sets the visible/hidden state of an item.
"value" is an integer, where 0=hidden and 1=visible.

toggleVisible()
Toggles the visible/hidden state of an item.

TOC

About: A representation of a complete table of contents. A toc may contain any number of items.

Properties:

title – a string identifier. Used as the title of the legend.
name – a string identifier. Used as label of the root GROUP.
root – the top-level GROUP object containing all other items.
IsInited – a boolean value used internally to flag if the toc has yet been initialized.
divToc – a reference to a DIV element where the toc is written
divTocHelp – a reference to a DIV element where the help is written
autoRefreshMap – a boolean indicating whether changes to the toc trigger a map refresh

Methods:

TOC(title,name,autoRefreshMap)
Constructor. Returns an object of class TOC.
"title" is a string containing the text to display at the top of the page.
"name" is a string containing the name of the toc, also used to label the root GROUP.
"autoRefreshMap" is a boolean indicating whether to automatically generate map requests when the toc changes in a way that would affect the map display. Note that there may be performance considerations to take into account when setting this value to true.

init()
Initializes internal values. Called internally.

setOutput(divToc,divTocHelp)
Sets the DIV elements used for output.
Must be called before the TOC is generated.
If either element is null then that portion of the toc will not be written.

addItem(item)
Adds an item to the root GROUP. Returns the item.
"item" is a reference to a GROUP or LAYER object.

addGroup(group)
addLayer(layer)

Deprecated methods for backward compatibility.
Functionally equivalent to addItem(item)

onFolderClick(tocid)
An event handler.
Called when user clicks on the open/closed folder icon.
Default action is to toggle the open/closed state of the folder.
"tocid" is an integer that uniquely identifies an item within the toc.

onLayerClick(tocid)
An event handler.
Called when user clicks on the layer icon.
Default action is set the active state of the layer to true.
"tocid" is an integer that uniquely identifies an item within the toc.

onVisibleClick(tocid)
An event handler.
Called when user clicks on the visibility checkbox icon.
Default action is to toggle layer visibility, or cycle-set group visibility.
"tocid" is an integer that uniquely identifies an item within the toc.

onActiveClick(tocid)
An event handler.
Called when user clicks on the active radio icon.
Default action is to set the active state of a layer to true, no effect on groups.
"tocid" is an integer that uniquely identifies an item within the toc.

onSwatchClick(tocid)
An event handler.
Called when user clicks on the swatch icon.
There is no default action, developer must implement desired behaviour.
"tocid" is an integer that uniquely identifies an item within the toc.

onNameClick(tocid)
An event handler.
Called when user clicks on the group or layer name text.
Default action is set layer active, or toggle group open/closed state.
"tocid" is an integer that uniquely identifies an item within the toc.

onLabelClick(tocid)
An event handler.
Called when user clicks on the label toggle icon.
Default action is call ITEM.toggleLabel().
However, toggleLabel() must be implemented by the developer.
"tocid" is an integer that uniquely identifies an item within the toc.

refresh()
Intended for internal use only.
Refreshes the toc display.

refreshMap()
Intended for internal use only.
Issues a map request if appropriate.

writeHTML()

Returns a string containing the contents of the toc in HTML format.
Will cascade and call writeHTML() for each item object in the root group.

writeHelpHTML()
Returns a string containing a set of instructions in HTML format.

Note: Do not directly call any of the "free-standing" versions of the class methods. For example, "toc.writeHTML()" would be a valid call, but directly calling "TOC_writeHTML();" would be an error.

Other Code:

var _nextTocID
a global integer that maintains the next available unique identifier.
Do not alter.

function getNextTocID()
returns the next available unique identifier.
Called internally.

Acknowledgements

Thanks to Andrew Eddie (A.Eddie@toowoomba.qld.gov.au) for bouncing around ideas and sharing his modifications to v.1.2 (some of which are incorporated here in one way or another) and for providing me with a bit of incentive for finishing off this latest version. :-D

Thanks to all others who may have contributed ideas, suggestions, code, bug reports, etc.

History

(yanked from dbgtCode.js to reduce size)

```
// version 1.0 05/16/2002
//
// version 1.1 changes:  (only released as a patch)
//   added simple rescrolling after a refresh
//
// version 1.2 11/26/2002 changes:
//   additional testing with ArcIMS 4.0
//   object.write* methods no longer write directly to
//     a document, instead simply return an html string,
//     method names changed (postfixed with "HTML")
//   output is inteded to go to DIV's, thus fixing the
//     flicker and scrolling problems in previous version
//   toc.setOutput() method added
//   .divToc and divTocHelp added to TOC object
//   toc.refresh() method rewritten
//   easier to set optional help, simply do (or do not) supply
//     a DIV for the help when calling toc.setOutput()
//   (above changes also improve support for toc in a popup window)
```

```
//   layer info button removed, now a hyperlink around layer name
//   .iconInfo removed from RESOURCE object
//   added a sample legend image for sf demo
//   added comments inside writeHTML() methods to aid customizers
//   added these comments regarding table structure to aid customizers:
//
//     the toc table is 5 columns wide as so:
//
//     group rows:
//       | fldr | visi | groupname               |
//     layer rows:
//       | bars | visi | actv | swch | layername |
//     legend rows:
//       | emty | emty | emty | legend           |
//
//
// version 1.21 12/11/2002 changes:
//   added additional formatting in LAYER_writeHTML() to keep cells aligned
//
// version 1.3 12/11/2002 - 01/23/2003 changes: (never publicly released)
//   changes in toc.htm to support NS7
//   additional testing w/ NS7
//   title of toc is now a property of the TOC object instead of RESOURCE object
//   added autoRefreshMap property to TOC and its constructor
//   added on*Click methods as easier hooks for customization
//   removed use of displayLayerInfoButton variable
//   added alt tags with short hints for icon images
//   added nowrap attribute to group & layer name td's
//   preliminary support for nesting - add layers to toc, add groups to groups
//   minor modifications to style sheet dbgtStyle.css
//   minor code reformatting
//
// version 1.4 01/26/2003 changes:
//   major restructuring and rewrites to support nested groups/layers
//   still "mostly" backwards compatible with previous versions of data file (dbgtData.js)
//   addition of ITEM template class
//   table is now a single column with ragged alignment (no longer 5 distinct columns)
//   additional on*Click event methods added, all user interaction now passes through
//     an on*Click method, makes customization much cleaner/easier
//   added "customization note:" sections within code to (hopefully) help answer frequent questions
//   standardized use of quotes/dquotes (quotes are javascript string delims, dquotes output in html)
//   cleaned up unused attributes of dbgtResource in dbgtData.js
//   added partial support for label toggling (developer must add in actual logic)
//   more Netscape tweaking
//
```

Finally

Feel free to send comments, suggestions, or bug reports. Officially, this is unsupported code, but unofficially, if there's something wrong with it or there's a useful addition, and I find the time to make the change, then I'll post updates as possible.